Graphics Performance
Benchmarking of Android Devices
Using OpenGL ES

Aram Akopian
4/30/2015
Contents

• Introduction
• Research overview
• Design and implementation
• Testing and results
• Conclusion
• Demo
• Q&A
• About me
  – Software Engineer with 5+ years experience in the Aerospace & Defense industry
  – Professional experience includes embedded software and desktop applications
  – Currently working in data visualization
  – B.S. Computer Science, CSUN ‘09
Research Objective

• **What**
  Benchmarking graphics performance of mobile Android devices

• **How**
  Android app featuring a suite of user-customizable benchmark tests written in OpenGL

• **Why**
  Evaluate the graphical capabilities of a device and demonstrate the benefits of user-customizable benchmarks
Motivation

• Diverse Android device market
• Rapidly changing devices and hardware advancements
• Rise in mobile gaming popularity
• Limited benchmark apps dedicated to 3D graphics
• Lack of user-customization in benchmark apps
Goals/Objectives

• Create a 3D graphics benchmarking app for Android
  – Provide a variety of test scenes
  – Decompose typical scene elements
  – Allow users to build the scene

• Provide performance results
  – Device assessment
  – Device-to-device comparison
Research Overview

Setup
- Technology and architecture
- Project environment

Technology
- Android graphics architecture
- OpenGL ES 2.0
- Mobile GPU architecture

Study
- Benchmarking concepts
- Trade Study

Develop
- Requirements
- App design
- Benchmark scene design

Test
- Devices under test
- Performance results
- Analysis

MOBILESpeed
Graphics Performance Benchmark
Technology and Architecture
Project Environment

• Steps to configure the project environment
  1. Install Java JRE/JDK 6+
  2. Install Eclipse
  3. Install Android Development Tools (ADT) plugin
  4. Configure Github repository
  5. Create a new Android project
  6. Setup an Android device for testing
Android Project in Eclipse

• Device support
  – Target SDK

• Project tree
  – src
  – res
    • drawable
    • layout
    • values
  – Manifest.xml

Eclipse project view
Project Testing Configuration

Option A
- Attach physical device
  - Connect via USB
  - Preferred method due to better performance

Option B
- Setup virtual device emulator
  - Slow, less stable
  - More device options
Android graphics architecture
OpenGL ES 2.0
Mobile GPU architecture

GRAPHICS TECHNOLOGY
• Supports 2D/3D graphics
  – 2D graphics: Canvas API
    • Menus, status bars, buttons
  – 3D graphics: OpenGL API
• All drawing is done to a graphics component called a Surface
  – GLSurfaceView
  – Buffer for graphics data
Surface rendering process

1. **Producer** – apps write data to graphics buffers
2. **Consumer** – a service reads and processes the buffers
3. **Composer** – performs buffer composition to create a displayable output

Data flow in the graphics pipeline
OpenGL ES 2.0

• **OpenGL** is a free, portable, and cross-language desktop graphics API

• **OpenGL ES** is a variation created for embedded systems
  – Reduced, more essential set of functionality
  – Programmable rendering pipeline

• MOBILESpeed uses API version 2.0
OpenGL ES 2.0 Rendering Pipeline

• An object to be drawn is specified as a set of vertices which are converted to pixels on a screen to represent that object.
Mobile GPU Architecture

• Key features of mobile GPUs
  – Differ from desktop GPUs architecturally
    • Immediate Mode Renderers vs. Tile-Based Renderers
  – Reduce wasteful processing (delay some actions)
  – Designed for lower power consumption, limited memory bandwidth

• These are important to understand when designing a benchmark app
Tile-Based Renderers (TBR)

- Optimize the drawing process by modifying steps in the rendering pipeline
  - Reorder draw commands
  - Defer fragment processing until necessary
- Divide the scene into tiles
Scene Tiling

• GPU creates a grid of tiles (NxN area of pixels)
• Work is performed on a per-tile basis
• Rendering is complete in multiple passes
• Advantages
  – Reduced bus bandwidth
  – Parallel design
• Disadvantages
  – Bad for complex geometry
Define Benchmark
Processing Bottlenecks
Frame Rate

BENCHMARKING CONCEPTS
Benchmarking Concepts

- **Benchmark**: a standardized problem or test that serves as a basis for evaluation or comparison
- Its purpose is to serve as a reference metric to which other measurements can be made and compared against
- Identify bottlenecks in the processing chain
Bottleneck describes a situation where the performance of a process is limited by some component, resulting in lower throughput.

Sources of bottlenecks:

- Hardware
  - Device component (CPU, GPU, memory) performance
  - May be mitigated by software

- Software
  - Poor implementation of logic, API, drivers, etc.
Frame Rate

- **Frame rate** is the speed at which an image can be refreshed or redrawn
- Most popular benchmark metric for video games
- Frame rate is measured as frame time or frames per second (FPS)
- MOBILESpeed utilizes frame time for device analysis

Frame time vs. frames per second
Android Benchmarking Apps

TRADE STUDY
Benchmark Apps for Android

• Research and analysis of available graphics benchmarking apps for Android was conducted

• Many benchmark apps were discovered, few specifically designed for graphics

• The features of these apps were considered during the design of MOBILESpeed
Requirements
Application Design
Benchmark Scene Design

DEVELOPMENT
A set of application requirements were written after a period of research and analysis of benchmarking applications.

Requirements were grouped into functional, interface, and performance areas.
Application Design

• A significant amount of design work was performed prior to development to create a quality software product that is intuitive and user-friendly

• Main focus areas:
  – Android software architecture and design patterns
  – User interface design
    • Building efficient UI with XML
  – Constraints

• Software architecture was modeled with UML
• UI mockups were created
Final Product Screenshots

App Home screen
Final Product Screenshots

App Tests screen
### Final Product Screenshots

**App Results screen**

<table>
<thead>
<tr>
<th>Test</th>
<th>Min Frame Time</th>
<th>Max Frame Time</th>
<th>Average Frame Time</th>
<th>Average FPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Triangles</td>
<td>10000</td>
<td>8.0 ms</td>
<td>28.0 ms</td>
<td>17.15 ms</td>
</tr>
<tr>
<td>Collision</td>
<td>10</td>
<td>8.0 ms</td>
<td>35.0 ms</td>
<td>17.14 ms</td>
</tr>
<tr>
<td>Texture</td>
<td>512x512</td>
<td>3.0 ms</td>
<td>35.0 ms</td>
<td>17.14 ms</td>
</tr>
<tr>
<td>Lines</td>
<td>10000</td>
<td>8.0 ms</td>
<td>66.0 ms</td>
<td>21.47 ms</td>
</tr>
<tr>
<td>Bus Bandwidth</td>
<td>160 MB</td>
<td>1321.0 ms</td>
<td>1383.0 ms</td>
<td>1346.07 ms</td>
</tr>
</tbody>
</table>
### Final Product Screenshots

**Model**: Acer A1-810

**OS Version**: Linux 3.4.67

**API Level**: 4.4.2 (SDK 19)

**CPU**: 4 cores | 1.2 GHz

**Memory**: 1.0 GB

**Display Resolution**: 1024x720 @58.3Hz | 160 dpi

**GPU**: PowerVR SGX 544MP

**3D Graphics**: OpenGL ES 2.0 build 1.12@2824438

- Max Texture Units: 8
- Max Texture Size: 4096x4096

**Supported Extensions**:
- GL_EXT_debug_marker
- GL_OES_rgb8_rgba8
- GL_OES_depth24
- GL_OES_vertex_half_float
- GL_OES_texture_float
- GL_OES_texture_half_float
- GL_OES_element_index_uint
- GL_OES_mapbuffer
- GL_OES_fragment_precision_high
- GL_OES_compressed_ETC1_RGB8_texture
- GL_OES EGL_image

---

**App Device Info screen**
Final Product Screenshots

App About screen
Benchmark Scene Design

• Created a set of 9 customizable benchmark test scenes
  – Points, Lines, Triangles
  – Texture, Lighting, Collision
  – Dynamic, Static, Bus Bandwidth

• Each test can be executed individually or combined with others

• The goal was to decompose the various elements of a typical scene in a game

• A study of benchmarking techniques and personal experience influenced the test scene designs
Benchmark Scenes

Points
Lines
Triangles
Texture
Lighting
Collision
Dynamic
Static
Bus Bandwidth
Combined
Device Testing

• 6 different Android devices of various classes were tested using MOBILESpeed
• All benchmark test scenes were executed both individually and combined
• Each device was tested at a normal and high intensity level
## Hardware Specs for Devices Tested

<table>
<thead>
<tr>
<th>Device</th>
<th>Graphics</th>
<th>Display</th>
<th>CPU</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acer Iconia A1</td>
<td>PowerVR SGX 544MP</td>
<td>7.9 inches</td>
<td>1.2 GHz</td>
<td>1.0 GB</td>
</tr>
<tr>
<td></td>
<td>OpenGL ES 2.0</td>
<td>1024x720</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>8 Texture units</td>
<td>58.3Hz refresh</td>
<td>4 cores</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4096x4096 Texture size</td>
<td>160 dpi</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Samsung Galaxy Tab 2</td>
<td>PowerVR SGX 540</td>
<td>10.1 inches</td>
<td>1.0 GHz</td>
<td>0.8 GB</td>
</tr>
<tr>
<td></td>
<td>OpenGL ES 2.0</td>
<td>1280x752</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>8 Texture units</td>
<td>60.0Hz refresh</td>
<td>2 cores</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2048x2048 Texture size</td>
<td>160 dpi</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HTC One V</td>
<td>Adreno 205</td>
<td>3.7 inches</td>
<td>1.0 GHz</td>
<td>0.4 GB</td>
</tr>
<tr>
<td></td>
<td>OpenGL ES 2.0</td>
<td>800x480</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>16 Texture units</td>
<td>60.0Hz refresh</td>
<td>1 core</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4096x4096 Texture size</td>
<td>240 dpi</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Hardware Specs for Devices Tested

<table>
<thead>
<tr>
<th>Device</th>
<th>Graphics</th>
<th>Display</th>
<th>CPU</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Samsung Galaxy Note 3</strong></td>
<td>Adreno 330</td>
<td>5.7 inches</td>
<td>2.3 GHz</td>
<td>2.4 GB</td>
</tr>
<tr>
<td></td>
<td>OpenGL ES 3.0</td>
<td>1920x1080</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>16 Texture units</td>
<td>60.0Hz refresh</td>
<td>4 cores</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4096x4096</td>
<td>480 dpi</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Texture size</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>HTC One</strong></td>
<td>Adreno 320</td>
<td>4.7 inches</td>
<td>1.7 GHz</td>
<td>1.8 GB</td>
</tr>
<tr>
<td></td>
<td>OpenGL ES 3.0</td>
<td>1920x1080</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>16 Texture units</td>
<td>60.0Hz refresh</td>
<td>4 cores</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4096x4096</td>
<td>480 dpi</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Texture size</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>LG G3</strong></td>
<td>Adreno 330</td>
<td>5.5 inches</td>
<td>2.5 GHz</td>
<td>2.8 GB</td>
</tr>
<tr>
<td></td>
<td>OpenGL ES 3.0</td>
<td>2392x1440</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>16 Texture units</td>
<td>60.0Hz refresh</td>
<td>4 cores</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4096x4096</td>
<td>640 dpi</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Texture size</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Performance Results & Analysis

**Results for Individual Normal Intensity Tests**

Frame Time in Milliseconds (lower is better)

- LG G3
- HTC One
- Samsung Galaxy Note 3
- HTC One V
- Samsung Galaxy Tab 2
- Acer Iconia A1

**Results for Individual High Intensity Tests**

Frame Time in Milliseconds (lower is better)

- LG G3
- HTC One
- Samsung Galaxy Note 3
- HTC One V
- Samsung Galaxy Tab 2
- Acer Iconia A1

Legend:
- Static (50k)
- Dynamic (8k)
- Collision (100)
- Lighting
- Texture (512)
- Triangles (20k)
- Lines - (20k)
- Points (20k)

- Static (125k)
- Dynamic (50k)
- Collision (1k)
- Lighting
- Texture (2048)
- Triangles (100k)
- Lines (100k)
- Points (100k)
Performance Results & Analysis

• Observations
  – Devices with superior hardware specifications perform better in most test cases
  – Degradation in performance is not proportional to the increase in object count
  – There is more diversity in frame time among test scenes for higher intensity tests
Future work
Review

CONCLUSION
Future Work

• Update existing test scenes
  – Allow broader range of user input values
  – Option for off-screen rendering
• Add more advanced test scenes
  – OpenGL driver overhead test
  – Multiple/complex fragment shader test
• Online database of performance results
• Release app on Google Play store
Conclusion

• Review
  – Android apps
  – Project environment
  – Graphics technologies
  – Benchmarking concepts
  – MOBILESpeed app
    • Requirements
    • Design
    • Development
  – Device performance benchmark and analysis

• Successfully created a user-customizable mobile graphics benchmarking Android application
• Demonstrated the benefits of benchmarking with MOBILESpeed
• Collected, presented and analyzed benchmark results for various Android devices
• Provided comprehensive material to create a benchmarking app
  – Design, development, tools, and techniques
Questions?

Demo